Loops and Arrays
Lecture 4 Assignments

Name: Khanne S. Labao
Student no.: 202209642
Course & year: BS in Computer Science

1. What is the output of the following program?

```
#include <stdio.h>

Int main (void)
{
    int i;

    i = 1;
    while (i <= 128) {
        print ("%d ", i);
        i *= 2;
    }

    return 0;
}
```

Ans: 1 2 4 8 16 32 64 128

2. Which one of the following statements is not equivalent to the other two (assuming that the loop bodies are the same)?

a.) while (i < 10) {. . .} b.) for (; i < 10;) { . . . } c.) do { . . . } while (i < 10);

```c
#include <stdio.h>

int main(void) {

    int i = 0;

    while (i < 10) {
            printf("%d ", i);
            i++;
    }

    printf("\n");

    i = 0;
    for (; i < 10;) {
            printf("%d ", i);
            i++;
    }

    printf("\n");

    i = 0;
    do {
            printf("%d ", i);
            i++;
    } while (i < 10);

    printf("\n");

    return 0;
}
```

Output:

```
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
```

When we compare the outputs of the provided loops, we cannot determine which one is different from the other two. The output of printf functions will be the same for all loops. However, we can tell them apart by how the compiler handles the loop's specific syntaxes. The do-while loop checks the stopping condition at the end of the iteration, whereas the other two loops check it at the start.

3. Convert item 1 into an equivalent for statement. You can validate your answer by checking if the produced outputs by both the while and for statements are similar.

Ans:

```c
#include <stdio.h>

int main(void)
{
    int i;
    i = 1;
    for (i=1; i<=128; i*=2) {
        printf("%d ", i);

    }
    return 0;
}
```

Output:

```
1 2 4 8 16 32 64 128
```

4. Write a code that computes for the power of two.

Ans:

```c
#include<stdio.h>

int main() {
  long int p;
  int n;

  printf("\n n    2 to the n");
  printf("\n===  ============");

  p = 1;
  for (n = 0; n < 11; ++n) {
    if (n == 0)
      p = 1;
    else
      p = p * 2;
    printf("\n%2d   %8ld", n, p);
  }
  return 0;
}
```

Output:

```
 n    2 to the n
===  ============
 0           1
 1           2
 2           4
 3           8
 4          16
 5          32
 6          64
 7         128
 8         256
 9         512
10        1024
```

5. Write a program that displays a one-month calendar.

Ans:

```c
#include <stdio.h>

int main(void)
{
    int days, start_day, i, j;

    printf("\nEnter number of days in month: ");
    scanf("%d", &days);

    if (days != 28 && days != 29 && days != 30 && days !=31)
    printf("Invalid number. Please try again.");

    else
    {

    printf("Enter starting day of the week (1=Sun, 7=Sat): ");
    scanf("%d", &start_day);
    printf("\n");

        if (start_day <= 0 || start_day >= 8)
        printf("Invalid number. Please try again.");

        else
        {
    /* prints the blank days of the first week */
        for (i = 1; i < start_day; i++)
            printf("   ");

        /* prints the calendar numbers */
                for (j = 1; j <= days; i++, j++) {
                printf("%3d", j);
                if (i % 7 == 0)
                    printf("\n");
                }

            printf("\n\n");
        }
    }
    return 0;
}
```

Output:

```
Enter number of days in month: 31
Enter starting day of the week (1=Sun, 7=Sat): 3

       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

6.

a. Revise line 16 such that you use a designated initializer to set pathways 0 and 2 to true, and the rest will be false. Make the initializer as short as possible.

```c
#include <stdio.h>
#include <stdbool.h>

#define NUM_PATHWAYS ((int) (sizeof(pathway) / sizeof(pathway[0])))

int main(){
    bool pathway [8] = {[0]= true,[2] = true};

    for (int i = 0; i < NUM_PATHWAYS; i++){

        if (pathway[i]){
            printf("pathway [%d] is open \n", i);
        }else{
            printf("pathway [%d] is close \n", i);
        }
    }
    return 0;
}
```

Ans: **bool pathway [8] = {[0]= true,[2] = true}**

b. Revise line 16 such that the initializer will be short as possible (without using a designated initializer)

```c
#include <stdio.h>
#include <stdbool.h>

#define NUM_PATHWAYS ((int) (sizeof(pathway) / sizeof(pathway[0])))

int main(){
    bool pathway [8] = {true, false, true};

    for (int i = 0; i < NUM_PATHWAYS; i++){
        if (pathway[i]){
            printf("pathway [%d] is open \n", i);
        }else{
            printf("pathway [%d] is close \n", i);
        }
    }

    return 0;
}
```

Ans: **bool pathway [8] = {true, false, true};**

7.

As a programming assignment:

1. Declare and initialize a road_networks multidimensional array that represents the adjacency matrix
2. Display the adjacency matrix. Put a bracket to the points/destinations that are considered as charging stations, e.g. [c], [d]
3. Given a point / destination, determine the nearest charging station. For example, if you are in point a, the nearest charging station is point c. If you are in point e, the nearest charging station is point d.
4. Bonus: Use a macro to define the size of the 2d array.

```c
#include <stdio.h>
#include <stdbool.h>
#include <ctype.h>

#define N 9

int main() {
    bool road_networks[N][N] = {
        {1, 1, 0, 0, 0, 1, 0, 0, 0},
        {1, 1, 1, 0, 0, 0, 0, 0, 0},
        {0, 1, 1, 0, 1, 1, 0, 0, 1},
        {0, 0, 0, 1, 1, 0, 0, 0, 0},
        {0, 0, 0, 1, 1, 0, 0, 0, 0},
        {1, 0, 1, 0, 0, 1, 0, 0, 0},
        {1, 0, 0, 1, 0, 0, 1, 0, 0},
        {0, 0, 0, 0, 0, 1, 0, 1, 1},
        {0, 0, 0, 0, 0, 0, 0, 1, 1}
    };

    //display the adjacency matrix
    printf("Adjacency Matrix:\n");
    printf("\ta\tb\t[c]\t[d]\te\tf\tg\th\ti\n");
    for (int i = 0; i < N; i++) {
        switch(i) {
            case 2:
            printf("[c]\t");
                break;
            case 3:
            printf("[d]\t");
                break;
            default:
                printf("%c\t", 'a' + i);
        }
        for (int j = 0; j < N; j++) {
            printf("%d\t", road_networks[i][j]);
        }
        printf("\n");
    }
    printf("\n");

    int start;
    printf("Which point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H, 8 - I\n\n");
    scanf("%d", &start);
```

```c
    //convert to uppercase letter
    char startChar = toupper(start + 'a');

    //cases for each station
    switch(start){
        case 0:
            printf("At point: %c", startChar);
            printf("\nC arrived to charging station.");
        break;
        case 1:
            printf("At point: %c", startChar);
            printf("\nC arrived to charging station.");
        break;
        case 2:
            printf("At point: %c", startChar);
            printf("\nC is a charging station.");
        break;
        case 3:
            printf("At point: %c", startChar);
            printf("\nD is a charging station.");
        break;
        case 4:
            printf("At point: %c", startChar);
            printf("\nD arrived to charging station.");
        break;
        case 5:
            printf("At point: %c", startChar);
            printf("\nC arrived to charging station.");
        break;
        case 6:
            printf("At point: %c", startChar);
            printf("\nD arrived to charging station.");
        break;
        case 7:
            printf("At point: %c", startChar);
            printf("\nNo nearby charging station.");
        break;
        case 8:
            printf("At point: %c", startChar);
            printf("\nNo nearby charging station.");
        break;
        default:
        printf("The station does not exist.");
    }
}
```